Rozdział 2. Algorytmika i programowanie w języku Python

Temat 6. Algorytmy na tekstach

3 godziny lekcyjne

Na lekcji wykorzystasz:  Podręcznik  Prezentacja  Plik

Cele ogólne

Uczeń:

- planuje kolejne kroki rozwiązywania problemu, z uwzględnieniem podstawowych etapów myślenia komutacyjnego (I.1),
- stosuje przy rozwiązywaniu problemów algorytmy poznane w szkole podstawowej (I.2) oraz algorytmy na tekstach: porównywanie tekstów, wyszukiwania wzorca w tekście metodą naiwną (I.2b),
- sprawdza poprawność działania algorytmów dla przykładowych danych (I.5),
- projektuje i programuje rozwiązania problemów z różnych dziedzin, stosuje przy tym instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje z parametrami i bez parametrów, testuje poprawność programów dla różnych danych (II.1),
- do realizacji rozwiązań prawidłowo dobiera środowisko programistyczne (II.2).

Cele szczegółowe

Zapamiętanie

Uczeń:

- definiuje pojęcia: kod liczbowy znaku, emoji, tablica znaków UNICODE, tablica ASCII,
- do przetwarzania tekstów w języku Python stosuje typ i klasę *str*.

Zrozumienie

Uczeń:

- omawia sposoby zapisywania informacji tekstowych w komputerze,
- definiuje dostęp do pojedynczego znaku łańcucha,
- stosuje algorytmy przetwarzania, porównania i wyszukiwania tekstów,
- wyjaśnia działanie funkcji `chr`, `len` oraz metod `find` i `append`.

Zastosowanie

Uczeń:

- stosuje funkcje wypisujące kod ASCII symbolu i symbol kodu ASCII,
- używa w implementacjach typ tekstowy `str`, funkcji `chr` i `len` oraz metod `find` oraz `append` przy przetwarzaniu tekstów w języku Python,
- stosuje konkatenację łańcuchów,
- wyszukuje wzorzec w tekście.

Tworzenie

Uczeń:

- implementuje programy wyświetlające tablicę kodów ASCII,
- tworzy programy porównujące znaki, łańcuchy znaków, w tym ich rozmiar,
- implementuje programy modyfikujące łańcuchy, wyszukujące wzorce w tekście.

Środki dydaktyczne:

- komputer z dostępem do internetu,
- projektor multimedialny lub tablica multimedialna,
- edytor Mu,
- pliki dla ucznia w serwisie informatyka.edu.pl:
    1. *wykreslanka.xlsx*
    2. *wzorzec_w_tekscie.py*
    3. *wzgorze.txt*
    4. *emoji.pdf*
- materiał multimedialny dostępny na stronie dlanauczyciela.pl:
    1. *Kody ASCII i Unicode w języku Python*
    2. *Wyszukiwanie wzorca w tekście*

Metody pracy:

- dyskusja (MD),
- burza mózgów (MBM),
- ćwiczenia, zadania (MC),
- praca z wykorzystaniem programów komputerowych i internetu (MEL),
- praca z podręcznikiem (MPP).

Wiedza uprzednia ucznia

Uczeń:

- formułuje i zapisuje w postaci algorytmów polecenia składające się na: rozwiązanie problemów z życia codziennego i z różnych przedmiotów,
- stosuje podstawowe instrukcje języka Python – wprowadzanie i wypisywanie danych (`input`, `print`), instrukcje warunkowe i iteracyjne, stosuje o wcięciach w kodzie źródłowym,
- testuje na komputerze swoje programy pod względem zgodności ze specyfikacją oraz je poprawia i optymalizuje, objaśnia przebieg działania programów,
- stosuje przy rozwiązywaniu problemów podstawowe algorytmy porównywania i wyszukiwania.

Wskazówki do prowadzenia lekcji

Wskazówki ogólne

1. Temat proponujemy podzielić w następujący sposób:
   Lekcje 1 i 2: zagadnienia 6.1–6.3
   Lekcja 3: zagadnienie 6.4
2. W temacie uczniowie przypomną sobie poznane wcześniej sposoby implementacji algorytmów w języku Python, pojęcie funkcji, klasy, metody oraz strukturę programu, w tym znaczenie wcięć.
3. W pierwszej części zajęć warto się skupić na uświadomieniu uczniom znaczenia narzędzi do przetwarzania tekstu i konsekwencji błędów językowych.
4. Przy omawianiu liter i znaków jako liczb warto wspomnieć o interpretacji informacji w komputerze (kod binarny), a następnie wykonać ćw. 1 ze s. 96, co pozwoli na płynne przejście do omówienia znaków ASCII i tablicy Unicode (tabela 6.2, s. 97) i typu danych `str`.
5. Dokładna analiza programu *Znaki* (s. 98) pomoże uczniom lepiej zrozumieć zagadnienie kodów liczbowych i zastosowanie funkcji `chr`. Przy tej okazji można przypomnieć uczniom budowę i działanie funkcji `for`. Warto w tym celu wykonać i omówić również ćw. 2 i 3 (s. 98 i 99).
6. Do wprowadzenia funkcji `ord` można wykorzystać modyfikację programu *Znaki* – *Znaki 2* (s. 99). Jest to też dobry moment na przypomnienie, jak działa pętla `while`. Wykonanie ćw. 4 ze s. 99 pomoże uczniom to zrozumieć i zapamiętać.
7. Przy omawianiu łańcuchów znaków warto zwrócić uwagę uczniów na praktyczną stronę zagadnienia, wykorzystując program *Porównywanie adresów* (s. 100) oraz wykonując ćw. 5 ze s. 101. Możemy je wykorzystać do wprowadzenia pojęć: stała, scalanie (konkatenacja) napisów.
8. Warto przypomnieć uczniom budowę i zastosowania list w języku Python oraz związane z nimi pojęcie indeksu (w tym indeksów ujemnych). Dostęp do znaków łańcucha, podobnie jak do struktury listy, obrazuje rys. 6.3 (s. 102).
9. Program *Sprawdzenie adresu* (s. 104) jest dobrym momentem na przypomnienie uczniom celowości i sposobów korzystania z własnych funkcji oraz ich struktury. Można również przypomnieć uczniom formuły logiczne. Wykonanie ćw. 6 ze s. 105 pozwoli przekonać się, czy uczniowie dobrze opanowali materiał.
10. Program *Sprawdzenie adresu* można zmodyfikować, wprowadzając metody klasy `str` – `find` i `rfind`. Warto przeanalizować kod źródłowy funkcji *CzyPoprawnyAdres* z wykorzystaniem wymienionych metod na s. 106.
11. Wyszukiwanie duplikatów w tekście jest istotną operacją w pracy analityka danych. Warto przeanalizować działanie programu *Usuwanie powtórzeń* i wykonać ćw. 7 i 8 (s. 108). Wprowadzamy przy okazji kolejną metodę – `append`.
12. Zagadnienie wyszukiwania wzorca w tekście warto dokładnie przeanalizować i omówić z uczniami podany algorytm. Jest to też dobry moment na przeprowadzenie dyskusji, dlaczego ten algorytm nazywamy naiwnym.
13. W wypadku pracy zdalnej można skorzystać ze środowiska Google Colaboratory.

Wskazówki dla klas mniej zaawansowanych

1. We wprowadzeniu warto odwołać się do zastosowań praktycznych: autokorekty w edytorze tekstu, wyszukiwarek internetowych czy formularzy online, aby przybliżyć zagadnienie oraz pokazać jego rangę.
2. Warto przeanalizować i wykonać z uczniami wszystkie ćwiczenia z tego tematu.
3. Typowe błędy i trudności w realizacji tematu:
    - brak motywacji do nauki programowania, wynikający z nieświadomości uczniów, że ta umiejętność przydaje się w wielu dziedzinach,
    - niski poziom umiejętności matematycznych, problemy z logicznym myśleniem – warto dokładnie analizować z uczniami podane kody źródłowe,
    - słaba znajomość podstawowych konstrukcji języka Python z poprzednich etapów nauczania – warto wówczas odwołać się do dodatków w końcowej części podręcznika.

Wskazówki dla klas bardziej zaawansowanych

1. Warto zachęcić uczniów do rozwiązania zadań o podwyższonym stopniu trudności, czyli oznaczonych trzema gwiazdkami.
2. Uczniowie mogą modyfikować napisane wcześniej kody źródłowe, szukając innych rozwiązań tych samych sytuacji problemowych.
3. Można poprosić uczniów o znalezienie informacji na temat algorytmu Karpa–Rabina (omówionego w podręczniku *Informatyka na czasie 3. Zakres rozszerzony*) lub Knutha–Morrisa–Pratta, zastosowań wybranego algorytmu, a nawet jego implementację.
4. Uczniom zainteresowanym algorytmiką i programowaniem można zasugerować utworzenie konta w portalu szkopul.edu.pl.
5. Na podstawie klasy `str` można wprowadzić pojęcia: klasy, obiektu, metody, języka obiektowego.

Przykładowe rozwiązania oraz komentarze do wybranych ćwiczeń i zadań

(ćw. 1, s. 96) Rozwiązanie w plikach: *T6_CW1a*, *T6_CW1b*.

(ćw. 2, s. 98) Rozwiązanie w pliku *T6_CW2*.

(ćw. 3, s. 99) Rozwiązanie w pliku *T6_CW3*.

(ćw. 4, s. 99) Rozwiązanie w plikach: *T6_CW4a*, *T6_CW4b*.

(ćw. 5, s. 101) Rozwiązanie w pliku *T6_CW5*.

(ćw. 6, s. 105) Rozwiązanie w plikach: *T6_CW6a*, *T6_CW6b*.

(ćw. 7, s. 108) Rozwiązanie w pliku *T6_CW7*.

(ćw. 8, s. 108) Rozwiązanie w pliku *T6_CW8*.

(ćw. 9, s. 109) Plik *wykreslanka.xls* do pobrania ze strony informatyka.edu.pl.

(ćw. 10, s. 110) Rozwiązanie w plikach: *T6_CW10a*, *T6_CW10b*.

(ćw. 11, s. 112) Plik *wzorzec_w_tekscie.py* do pobrania ze strony informatyka.edu.pl. Rozwiązanie w pliku *T6_CW11a*.

(zad. 1) Rozwiązanie w pliku *T6_ZAD1*.

(zad. 2) Rozwiązanie w pliku *T6_ZAD2*.

(zad. 3) Rozwiązanie w pliku *T6_ZAD3*.

(zad. 4) Rozwiązanie w pliku *T6_ZAD4*.

(zad. 5) Rozwiązanie w pliku *T6_ZAD5*.

(zad. 6) Rozwiązanie w plikach: *T6_ZAD6_v1*, *T6_ZAD6_v2*, *T6_ZAD6_v3*, *T6_ZAD6_v4*.

(zad. 7) Plik *emoji.pdf* do pobrania ze strony informatyka.edu.pl. Modyfikacją programu może być wyświetlanie znaków między innymi symbolami. Rozwiązanie w pliku *T6_ZAD7*.

(zad. 8) Modyfikacją programu może być zliczanie liczby spółgłosek. Rozwiązanie w pliku *T6_ZAD8*.

(zad. 9) Plik *wzgorze.txt* do pobrania ze strony informatyka.edu.pl. Rozwiązanie w pliku *T6_ZAD9*.

(zad. 10) Rozwiązanie w pliku *T6_ZAD10*.

(zad. 11) Rozwiązanie w pliku *T6_ZAD11*.

(zad. 12) Rozwiązanie w plikach: *T6_ZAD12_v1*, *T6_ZAD12_v2*.

(zad. 13) Rozwiązanie w pliku *T6_ZAD13*.

(zad. 14) Rozwiązanie w pliku *T6_ZAD14*.

(zad. 15) Rozwiązanie w pliku *T6_ZAD15*.

(zad. 16) Rozwiązanie w pliku *T6_ZAD16*.

(zad. 17) Rozwiązanie w pliku *T6_ZAD17*.

(zad. 18) Rozwiązanie w plikach: *T6_ZAD18_v1*, *T6_ZAD18_v2*, *T6_ZAD18_v3*.